TITLE OF THE INVENTION

NON-INVASIVE ESTIMATION OF ROUND TRIP TIME IN A PACKET-ORIENTED ACKNOWLEDGE-BASED TRANSMISSION SYSTEM

BACKGROUND OF THE INVENTION

The present invention relates to protocol analysis, and more particularly to a non-invasive estimation of round trip time in a packet-oriented acknowledge-based transmission system using a probe at any point in the system to monitor the signals transmitted between a sender and a receiver.

Currently TCP/IP (Transmission Control Protocol/Internet Protocol) is used extensively in Intranet and Internet networks. With new wireless networks, especially the 2.5G and 3G generations, TCP/IP will be the protocol most used worldwide in telecommunications environments. Also TCP/IP should substitute for other protocols that are currently largely used in telecommunications infrastructures. However this large usage of TCP/IP requires the introduction of new measurement processes to characterize the behavior of this protocol and the quality of service (QoS) offered by applications using it. Some characteristics of this protocol should have an impact on the different applications using TCP/IP. A very important parameter is "round trip time" (RTT) that gives a measure of the delay perceived by a user or application sending a packet and receiving a response for it. This parameter is influenced by problems such as queuing delays in the network, erroneous dimensioning or configuration on the part of the network, bottlenecking, etc.

TCP/IP is a protocol that allows reliable data transfer between two terminal hosts. In order to provide reliability, packets that correctly reach the receiver are acknowledged and the sender is informed of the correct delivery. The time that elapses between the sending of the packet and the response provided by the acknowledgment is the RTT. Typically RTT is a measure of "how far" the receiver is from the sender, and gives good information regarding network behavior and quality perceived by users. It is clear that the evaluation of RTT could be performed easily at a sender side, but this point of measure is normally outside the scope of a network operator.

Fig. 1 (upper part) shows a graphic illustrating what is meant by RTT. As shown TCP/IP packets **14** travel from the sender **10** to the receiver **12** and the related acknowledgments **16** follow the inverse path.

To provide better throughput TCP/IP uses a sliding-window transmission algorithm. The sender periodically evaluates a value called *congestion window (cwnd)*. This parameter represents the number of packets transmitted and not already acknowledged. Its value is limited to an *advertised window (advwnd)* which represents the maximum buffer size used by the receiver.

GPRS (General Packet Radio Service) networks, as a mobile example of a TCP/IP network, are more sensitive to RTT than current Internet networks. This is due to the impact of the radio path on the total TCP/IP path and on the inter-working of the IP environment with the telecommunications switch environment. Fig. 2 shows a typical structure of a GPRS network. A SGSN (Switched General Signal Network) may be defined as the front end

between IP protocols and traditional telecommunications protocols, even if TCP/IP is used in a transparent way from the sender (Internet server) to the receiver (GPRS mobile).

GPRS networks provide new access to services over the IP suite. In order to evaluate the quality perceived by users, mobile operators perform a continuous monitoring of the involved devices and lines. Moreover users may access different services not necessarily related to the mobile operator itself. For these reasons SLAs (Service Level Agreements) are stipulated between different network owners, and a continuous monitoring of network behavior ensures the respect of such agreements. Therefore a mobile operator has to be able to perform RTT measures and to separate delays due to its network and to external ones.

If it is possible to monitor TCP traffic at the sender side, RTT evaluation is a simple task. In this case RTT is estimated by calculating the time elapsed between sending a packet and receiving the corresponding acknowledgment. Usually TCP senders evaluate RTT using this simple technique in order to compute the Retransmissioin Time Out (RTO – see Request for Comments RFC723 p. 41). This technique has some advantages: it is easy to implement and a non-invasive implementation may be realized at the sender side. Its disadvantages are that it can only be applied at the sender side and measurements cannot be performed in the presence of high rates of packet loss. An invasive technique, such as "ping", uses the same technique. This latter solution is well known and widely used in the Internet and other IP networks. In any event it cannot be implemented

in order to evaluate network response times perceived by single users. This is due to the fact that packets used by "ping" are typically smaller than the ones used in data transfers. Moreover "ping" does not use TCP, but ICMP (Internet Control Message Protocol).

To perform an RTT evaluation at an intermediate point, another known technique is based on TCP SYN and SYN acknowledgment measurement. This technique evaluates the time delay between particular TCP messages occurring only in certain cases in a TCP connection to setup the connection itself. This solution has the advantage of simplicity as it can perform a single RTT evaluation in a non-invasive manner at any intermediate point. However it cannot perform a continuous monitoring of network performance – it is not possible to track RTT variation during a particular connection. The single evaluation is performed monitoring packets smaller than the ones involved during data transfer and the sender delay is included in the RTT estimation.

What is desired is a method that allows continuous monitoring of network performance and tracking of RTT variation, especially during a particular connection.

BRIEF SUMMARY OF THE INVENTION

Accordingly the present invention provides a method of non-invasive evaluation of round trip time (RTT) between a sender and a receiver in a packet-oriented acknowledge-based telecommunications system that delivers mean values of RTT for groups of packets during a particular connection by continuous monitoring and following of possible variations due to network

congestion and radio link delay. The RTT is evaluated by analyzing a group of acknowledgments and their corresponding transmitted packets, instead of analyzing a single acknowledgment and its corresponding packet. A correlation between the acknowledgments and their packets is done looking forward and backward in time. Acknowledgments are sent by the receiver as soon as packets are delivered, while packets are transmitted by the sender after an acknowledgment has arrived. The total RTT is defined by the sum of two different components: the round trip time from a probe to receiver, RTTpr, and the round trip time from the probe to sender, RTTps, where the probe may be coupled to any interface point in the system. For a certain group of acknowledgments RTTpr is evaluated considering the corresponding packets backward in time, and RTTps is evaluated considering the corresponding packets forward in time. The sum of RTTpr and RTTps provides the estimation of RTT. As a result the RTT evaluation is non-invasive and may be performed at any intermediate point as well as at the receiver side, and RTT variations and congestion window variations may be tracked during a particular connection.

The objects, advantages and other novel features of the present invention are apparent from the following detailed description when read in conjunction with the appended claims and attached drawing.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING

Fig. 1 is a block diagram view of a telecommunications system for illustrating round trip time (RTT) as well as the evaluation of RTT according to the present invention.

Fig. 2 is a block diagram of a GPRS application and possible locations for monitoring probes according to the present invention.

Figs. 3a and 3b are a timeline diagram view and an extract of the timeline diagram view according to the present invention.

Fig. 4 is a diagrammatic view illustrating the estimation of RTT according to the present invention.

Fig. 5 is a diagrammatic view illustrating timestamp distance computation according to the present invention.

Fig. 6 is a block diagram view for the estimation of RTT according to the present invention.

Fig. 7 is a graphic view illustrating minimums validation according to the present invention.

Fig. 8 is a block diagram view illustrating estimation of a sliding window according to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

As indicated above, the upper portion of Fig. 1 illustrates RTT evaluation according to the prior art from the sender side. In the present invention a probe **18** is coupled to point of measure inside a network, which may be any accessible intermediate point such as gateways or typical network interfaces. The detemination of the RTT is performed in two steps: by determining the RTT probe-receiver (RTTpr) and the RTT probe-sender (RTTps) which are then added to determine the entire RTT. Fig. 2 shows examples for the location of the probe **18** at typical GPRS interfaces, such as Gi, Gb and Gn.

Each TCP connection may be described through a timeline diagram, like the one shown in Figs. 3a, 3b. Time increases in the downward direction and the left and right side vertical lines represent respectively the sender **10** and receiver **12** involved in the connection. TCP packets transmitted from sender to receiver are shown with thick arrows, and acknowledgments are shown with dashed arrows going from the receiver to sender. The TCP connection is probed at an intermediate point where captured packets and acknowledgments travel respectively later and sooner than at the sender side, i.e., a packet starts from the sender **10** at time t_1, passes the probe **18** at time t'_1 and reaches the receiver at time t_2. Also at time t_2 an acknowledgment automatically starts, passes the probe at time t'_2 and reaches the sender at time t_3. The sender is allowed to free new packets depending on the congestion window currently in use, according to the TCP sliding window mechanism, assuming the sender always is filled with packets to be sent, and the packet freed in response to the acknowledgment received at time t_3 passes the probe at time t'_3.

RTT is equal to $t_3 - t_1$ and is evaluated at the probe by estimating $t'_2 - t'_1$ and $t'_3 - t'_2$ that represent respectively RTTpr and RTTps. As shown in Figs. 3a and 3b t'_1, t'_2 and t'_3 are timestamps observed at the probe. What is needed is to find the correct sequence of packet, acknowledgment and new packet regardless of the sliding window algorithm implemented by the TCP sender. In fact other packets of the same connection flow between sender receiver during the analysis.

RTTpr is evaluated by two separate techniques:

- Timestamps (capture time at the probe) of a small group of acknowledgments, such as 4-5, are "moved" back in time in order to find the time sequence of packets that released them at the receiver. This operation is done by a distance function performed along the time series.
- Sequence numbers of the same group of acknowledgments are compared with the packets encountered during the described operation in order to perform a distance function along the sequence numbers themselves.

RTTps is evaluated in a similar way, searching which packets have been released at the sender by the acknowledgments considered. In this case a direct correspondence between sequence numbers doesn't exist and a congestion window estimator is used in order to find out. For protocols that do not use variable congestion windows or something comparable, the described method works well up to that point. However further improvement may be achieved for protocols that use variable congestion windows as follows.

In TCP connections acknowledgments are generated by the arrival of packets and packets are freed by the arrival of acknowledgments. First the timestamp procedure tries to find the most probable sequence of packets that has generated the group of acknowledgments considered. In this way RTTpr is evaluated. Likewise the same procedure is used to try to find the packets that have been released by the group of acknowledgments considered –

RTTps. The described algorithm performs a mean RTT evaluation for each group of acknowledgments within an acknowledgment time interval, called "Wack", captured at the probe. Typically these groups contain enough acknowledgments to validate the response. RTT is not stable during TCP connection, and it represents the degree of congestion experienced by the sender and receiver. To get better results and not to fail the measurement procedure, the algorithm does not consider too many acknowledgments for each mean RTT evaluation. To resolve these problems a trade-off is found by analyzing real data tracks and finding optimal working values. The algorithm performs each RTT evaluation considering a small time window containing a few acknowledgments. The number of acknowledgments considered preferably is more than 4 and less than 7. The time window may be chosen with a typical value of 2 seconds. A maximum RTT for both directions is given to optimize performance. The worst case value for RTTmax is preferably 5 seconds.

Referring now to Fig. 4 the course in time of the packets (PKT) and the acknowledgments (ACK) are shown. The first acknowledgment considered was received at the probe at time t^*. Then the algorithm considers the packets arriving in time before and after t^* for this group of acknowledgments. To limit the number of packets to be analyzed, the interval in this example is limited to [t^* - RTTmax; t^* + RTTmax + Wack). Considering the acknowledgments associated with a Wack at time t^* (see Fig. 5), their timestamps are compared with packets between t^* - RTTmax and t^* + RTTmax + Wack. Several distances are computed in this interval "moving"

Wack with a resolution step *ris*. The left side of Fig. 5 corresponds to the probe-receiver segment and the right side corresponds to the probe-sender segment. "Moving" Wack from left to right means to add an offset to each timestamp of the acknowledgments considered.

To simplify the algorithm description the interval [t^*; t^* + RTTmax + Wack] is considered. In this time interval RTTps is evaluated. To determine RTTps the packets released by the acknowledgments contained in Wack need to be found. This is performed with a resolution step of *ris* moving the timestamps of the considered acknowledgments from t^* to t^* + RTTmax + Wack. At the *i*th iteration the used timestamp for the *n*th acknowledgment is:

t_moved_ack(n) = t_ack(n) + i*ris

where *n* is the acknowledgment considered, *t_ack(n)* is its timestamp, *t_moved_ack(n)* is the "moved" value and *i* is the index of the iteration. For each iteration *i* the algorithm computes the mean timestamp distance between the "moved" acknowledgments of Wack and the nearest packets. In pseudo-code these operations are:

```
for i = 0 to N = round(RTTmax/ris) {
        for n = 1 to N_ack {
          - find the packet nearest to t_moved_ack(n);
          - compute dt(n) as the absolute value of the difference
        between the      timestamp of that packet and t_moved_ack(n);
        }
        DT(i) = sqrt(sum(square(dt(n))))/N_ack
}
```

where *N_ack* is the number of acknowledgments in Wack. The first "for . . . to" loop means "for every resolution step", while the second "for . . . to" loop means "for every acknowledgment."

To avoid computation load the *sqrt* function may be omitted since the Minimum of *DT(i)* is the same as the Minimum of *sqrt(DT(i))*. However for an easier understanding it is left in the above equation as well as in corresponding equations below.

After all iterations are computed, *DT(i)* contains the mean timestamp distances for each *i*. If it is possible to find an i_1 such that $DT(i_1)$ is smaller than the *DT(i)*'s for all other *i*'s, then

$$\text{RTTps} = i_1 * \text{ris}$$

RTTpr is evaluated in the same manner considering the time interval [t^* - RTTmax; t^*]. Indices are changed slightly in order to search the packets that released the acknowledgments under analysis.

As described above RTTpr and RTTps are obtained searching the minimum values for *DT(i)*. Typically *DT(i)* has several minimums. To obtain better performance other information contained in packets is used.

In practice an acknowledgment has a sequence number obtained directly from the packet sequence number and the packet size. For the sake of simplicity a direct correspondence may be considered, namely that acknowledgments generated by packets arrived at the receiver side have the same sequence numbers as the packets that generated them. Thus a second distance may be evaluated with those values. Considering the above pseudo-code a new distance is added:

for n = 1 to N_ack {

- find the packet nearest to t_moved_ack(n);

- compute dt(n) [see above]

- compute ds(n) as the absolute value of the difference between the sequence number of that packet and the sequence number of the acknowledgement considered

}

DS(i) = sqrt(sum(square(ds(n))))/N_ack

This operation may be performed directly considering the left side of Fig. 5. In fact to evaluate RTTpr it is necessary to find acknowledgments and packets with the same sequence numbers.

Unfortunately the relationship between the sequence numbers of the considered acknowledgments and the sequence numbers of packets released at the sender side is related to the congestion window used by the TCP sender (right side of Fig. 5). In this case a congestion window estimator is implemented and the estimated value used as offset for the acknowledgments. In pseudo-code this results in:

compute the congestion Window for Wack.

for n = 1 to N_ack {

- find the packet nearest to t_moved_ack(n);

- compute dt(n) [see above]

- compute ds(n) as the absolute value of the difference between the sequence number of the acknowledgment considered plus the congestion window estimated;

}

DS(i) = sqrt(sum(square(ds(n))))/N_ack

To estimate RTTps correctly this procedure is implemented for the right side of Fig. 5.

For estimation of the congestion window the TCP sender continuously adapts the congestion window as a function of network congestion. As already stated *cwnd* represents the number of packets not already acknowledged by the receiver, and its upper bound is regulated by the advertised window, *advwnd*. This sliding window solution implemented by standard TCP improves throughput, but has to be adaptive to meet network congestion status. Different implementations of TCP senders try to adapt the congestion window using different techniques. Basically acknowledgments give information regarding packet loss and delay along the network. Using this information, different adaptation rules are used.

The described estimator tracks packets and acknowledgments at an intermediate point. It easily evaluates the number of packets not already acknowledged at the probe-receiver side. This number, called *apparent window*, is used as a minimum value for the congestion window estimated. In case of packet loss its value could overcome the congestion window. Therefore this lower bound cannot always be used.

The estimator follows simple rules of increasing and decreasing the window, and tracks the whole connection. A simple version containing the fundamental steps is described below:

```
for all acknowledgments {
    if the considered acknowledgment identifies a packet loss:
        then [MD phase]
            cwnd(t + 1) = cwdn(t)/2
        else [AI phase]
            cwnd(t + 1) = cwnd(t) + 1/cwnd(t)
}
```

where the MD phase (Multiplicative Decrease) is applied when packet loss is occurring and the AI phase (Additive Increase) is performed when acknowledgments with new sequence numbers are received.

The simple algorithm described here is introduced to explain the main blocks implemented. To solve some problems related to different TCP implementors and different link behaviors, some additional blocks are added. A complete outline of the algorithm is shown in Fig. 8. As shown all estimated values are clipped to the advertised window, the maximum allowable value. Another minimum value for the congestion window, *minwind,* is also used in the case of packet loss (duplicate *ack* detected). Typically *minwind* is set to 1 and is inserted in order to clip estimated values to a correct lower bound. At the beginning of a connection the estimator starts doubling the congestion window for each acknowledgment received (Slow Start Block) instead of executing the AI phase. This allows a closer estimation of *cwnd* during the beginning of the connection. In this case the sender increases rapidly its congestion window using the slow start algorithm (see Requests for Comments). As described above the limit provided by the *apparent window* is

not always reliable. Therefore its value is used only if it is not too far from the estimated value obtained – see the block inserted after the AI phase and the Slow Start. Mean RTT for the acknowledgments contained in Wack is evaluated calculating *DT* and *DS* in both directions. The estimated congestion window is used to calculate *DS* for the probe-sender side. Finally two minimums for *DT* are searched in order to obtain RTTpr and RTTps. Correct minimum positions for *DT* are obtained analyzing *DS*. Fig. 6 depicts the complete algorithm that is iterated in order to track RTT during a complete TCP connection. Fig. 7 shows an example of how *DT* minimums are validated using *DS*.

Finally the algorithm performs continuous monitoring of RTT and Congestion Window of TCP/IP connections and may be applied, as an example, to GPRS traffic. In this case possible access points are intermediate interfaces such as Gi, Gn and Gb. Distances along timestamps and sequence numbers are determined, using part of the information contained in TCP headers of acknowledgments and packets. The algorithm works equally well in ordinary IP networks as well as more complex networks, such as GPRS and UMTS (Universal Mobile Telecommunication System), and on different packet oriented transmission systems.